In the Specification

Kindly replace paragraphs [0002]-[0004] with the following:

Technical Field ~~of the Invention~~

[0002] ~~The invention~~ This relates to the area of managing persistent data of an entity, ~~e.g.~~ *e.g.*, a company. In particular, ~~the invention~~ this relates to the ~~follow-up~~ tracing of this persistent data in a database by a system for database management.

[0003] It is difficult for a company to guarantee ~~the follow-up~~ tracing of the development process of strategic persistent data because this ~~follow-up~~ tracing has several objective obstacles:

the asynchronous and collaborative nature of the development of the process,

the very demanding nature of the ~~follow-up~~ tracing for constituting a real guarantee: the presence of one weak link definitively compromises the reliability of any response,

the non-availability of generic solutions for taking charge of the traceability in the ~~soft-ware~~ software layers on the market at a satisfactory level of granularity: OS, DBMS (database management system), development language,

the very high cost of rewriting existing applications and the very high cost of taking explicit account of the traceability by each application.

[0004] WO 99/35566 discloses a process for the identification and ~~follow-up~~ tracing of the ~~developments~~ development of a set of software components. The process allows the recording of the components by their name and their version. This classification at the file level does not respond to the problem of saving traces of data in a continuous manner, that is, at each modification of this data. In particular, the process proposed is not suitable for tracing a database modified at each write access.

Kindly replace paragraphs [0010]-[0020] with the following:

Summary ~~of the Invention~~

[0010] ~~This invention relates to~~ I provide a process for organizing a digital database in a traceable form including modifying a main digital database by addition or deletion or modification of a recording of the main database, wherein modifying the main database includes creating at least one digital recording including at least unique digital identifiers of concerned recordings and attributes of the main database, a unique digital identifier of a state of the main database corresponding to the

modification of the main database, elementary values of attributes assigned via elementary operations without proceeding to store non-modified attributes or recordings, and addition of the concerned recording in an internal historical database composed of at least one internal historical table, and

reading the main database, wherein reading relates to any final or previous state of the main database and includes receiving or intercepting an original request associated with the unique identifier of a target state in proceeding to a transformation of an original request to construct a modified request for addressing the historical database including criteria of the original request and the identifier of the target state, and reconstruction of the recording or recordings corresponding to the criteria of the original request and to the target state, wherein the reconstruction includes finding elementary values contained in the recordings of the historical base and corresponding to the criteria of the original request to reduce requirements of storage capacity and processing times.

Brief Description of the Drawings

[0011] ~~The invention~~ My disclosure will be better understood with the aid of the following description, made purely by way of explanation, of an embodiment ~~of the invention~~ with reference made to the attached figures.

[0012] Fig. 1 schematically shows a classic communication architecture between an application and a database.

[0013] Fig. 2 schematically shows a communication architecture similar to that of Fig. 1 and comprising the elements necessary for the application ~~of the invention~~.

[0014] Fig. 3 schematically shows the different means for accessing a database organized in a traceable manner and provided with a system ~~in accordance with the invention~~.

Detailed Description

[0015] ~~This invention~~ My method eliminates disadvantages of the prior art by providing a process for the ~~follow-up~~ tracing of the development of the data in an architecture based on an DBMS, comprising:

materialization of the intermediate versions and data ~~streams~~ flows resulting from operations performed on the database as its development proceeds at the level of elementary granularity (recording by recording and attribute by attribute);

possibility of "rapid" reconstitution and retrieval of every original historical framework state of each data version and each operation (wherein the term "rapid" means "without perceptible additional time connected to the restoration");

comprising:

mechanisms for reconstituting the stream of causal ~~dependence~~ dependency (of the source-destination type) between the data concerned;

mechanisms for notifying the reappraisal of operations in the past in the case of the development of the input data;

mechanisms of re-execution;

and covering the following particular cases and extensions:

taking account of the structural development (development of ~~scheme~~ a schema);

taking account of the development of applications;

taking account of ~~applications~~ existing applications in a flexible architectural framework;

~~schemes~~ schemas of gradual development of an architecture on the scale of the company;

management of virtual versions (alternative families and parallel hypotheses).

[0016] ~~The invention~~ My method thereby permits the exploitation of the base data in accordance with the ~~successive~~ sequential versions while limiting the requirements of time and storage capacity and to authorize retrieval on the fly.

[0017] A customary step consists of recording ~~successive~~ sequential versions of databases, ~~e.g.~~ *e.g.*, in the form of periodic storing on a support such as a magnetic cartridge with the completeness of the database corresponding to the current version. The search for information requires the advance restoration of the entire base starting from the support corresponding to the corresponding backup, then the querying of the base restored in this manner. For bases of important data and such as those used in the banking system, the insurance system or management, the volume corresponding to a state can exceed a terabyte, a volume which it is advisable to multiply by the number of backed up states.

[0018] That solution is totally not adapted for use in real time. ~~The invention~~ My method responds to the technical problem of using large-volume databases in real time. To this end, the ~~invention~~ method concerns in its most general meaning a process for organizing a digital database in a traceable form comprising steps for the modification of a main digital database by the addition or

deletion or modification of a recording of the main base and of the reading steps of the main database, characterized in that

the step of modifying the main database comprises an operation of creating at least one digital recording comprising at least:

the unique digital identifiers of the concerned recordings and attributes of the main database,

a digital identifier of the state of the main database corresponding to this modification of the main database,

the elementary values of the attributes assigned to them via elementary operations without proceeding to store non-modified attributes or recordings,

the addition of this recording in an internal historization base composed of at least one internal historization table,

and in that the reading step relating to any final or previous state of the main database consists in receiving (or intercepting) an original request associated with the unique identifier of the state aimed at, in proceeding to a transformation of this original request to construct a modified request for addressing the historical database comprising the criteria of the original request and the identifier of the state aimed at, and the reconstruction of the recording or recordings corresponding to the criteria of the original request and to the state aimed at, which reconstitution step consists of finding the elementary values contained in the recordings of the historical database and corresponding to the criteria of the original request (to reduce the requirements of storage capacity and the processing times).

[0019] According to one aspect, such recordings of the historization database also contain references to other recordings of the internal database to specify the connections of dynamic ~~dependence~~ dependency of the source-destination type constituting the causal stream of the interferences between the data versions. This operation of modifying the main base is advantageously a logic operation and the operation of addition in the historization database consists of adding:

a recording identifying the state of the base corresponding to the logic operation,

as many recordings as parameters of the logic operation,

a recording for the possible result of the logic operation, and

specifying by cognateness the regrouping of operations from the elementary level of modification to the level of the transaction, passing the number of semantic levels necessary for the applications.

[0020] According to another aspect, the main database comprises one or several tables organizing the development links between the identifiers of the ~~successive~~ sequential and alternative states of the main base and intended to organize the recordings of the internal database.

Kindly replace paragraphs [0023]-[0027] with the following:

[0023] ~~The invention~~ My method also concerns an architecture for managing a database, characterized in that this application can bring about modifications in the entire state of the main base and give rise, in the instance of an attempt to modify a previous state, to the creation of new alternatives of digital development of the main database, whose data will be generated by the same internal historical database.

[0024] The ~~dependence~~ dependency links may serve as recovery criteria for the operations already carried out. The updatings carried out on the various branches can be integrated or merged into the framework of a new state "inheriting" these branches.

[0025] The cases of the development of the structure of the data of the main database may be treated as particular cases of the development of the data of this base. However, little of the structure/~~scheme~~schema of this main base is described in the manner cited for the data, as a dictionary.

[0026] The historical database may be explored and queried by applications via the native mode of the DBMS to obtain information such as, ~~e.g.~~ *e.g.*, all the historical values of an attribute and all the (dynamic) incidents of every updating and to navigate along the versions and the ~~streams of~~ dynamic ~~dependence~~ dependency flows in a classic manner in accordance with the querying language in force required by the DBMS.

[0027] The management of the persistent data of a company (or of an organization in the broad sense) is generally entrusted to a specific software also called a DBMS (database management system). Computer applications propose interactive ergonomic means to the users that are capable of visualizing and developing the data of the database of the company by communicating with the

DBMS. The main features of the architecture are recalled to position the framework of the process of ~~the follow-up of~~ tracing the development of the data and to fix its minimum vocabulary.

Kindly replace paragraph [0031] with the following:

[0031] The persistence manager also allows the definition, consultation and development of the data structure, also called a "data ~~scheme~~ schema." Thus, the tables can be defined, deleted or restructured. In the latter instance columns can be added or deleted. At times, it is even useful to change the domain of an attribute or of other analog characteristics, which can imply implicit or explicit conversion processes of the data concerned.

Kindly replace paragraphs [0033-[0046] with the following:

[0033] Finally, it is specified that a database represents a coherent state of the real world represented. The data of the base develop in surges released by events via the operations (insertion, updating or deletion) generally grouped by transactions. The latter are characterized by particular properties called ACID (atomicity, ~~coherence~~ consistency, isolation and durability) that guarantee a certain level of quality.

[0034] Ensuring the traceability of persistent data amounts to supplying means that permit the ~~follow-up~~ tracing upstream and downstream from the data development process.

[0035] The process of developing data is a generally non-predictable succession of executions of elementary operations that read, transform and write the data in a repeated manner giving rise most frequently to multiple and complex interferences that render their ~~follow-up~~ tracing difficult and frequently impossible. Ensuring the traceability of the process amounts to being capable of going back at every moment to the origins (beginnings) of the process, finding the values of the original data, being able to ~~follow~~ trace and understand their consequences during the course of the operations in terms of the impact of changes. In terms of quality of the information, traceability is very valuable because it allows the conformity of the result of an operation applied with the input data set to be guaranteed.

[0036] A classification of traceability is presented according to progressively more advanced levels to better understand the extent of its scope:

The first level of traceability, that can be qualified as elementary, is that of the representation and storage of data. It is therefore a matter of describing the structure, then of storing and identifying the data, whether it is a command, an article or even a mechanical component to be able to retrieve it later. This type of functionality is already ensured by specialized ~~soft-ware~~ software called database management systems (DBMS). The development process is manifested by the ~~successive~~ sequential application of elementary operations such as reading, insertion, updating and deletion. These elementary operations are generally grouped into transactions to maintain the ~~coherence~~ consistency of the data under conditions of competing use or of recovery in case of breakdown. At this level, updates have as a natural consequence the loss of existing values as a consequence of their replacement by new values since, by convention, only one data (with its attributes) can correspond to one identifier. This first level of traceability that is called elementary is indispensable but largely insufficient.

The second level of traceability authorizes a data to have several versions (distinct values) at the same time. This improves the traceability since it becomes possible to have values preceding as well as values following the execution of an operation or a process at any moment, which facilitates even more the comprehension of the development. The versioning introduces a valuable quality since the irreversibility can no longer be bypassed (the development of data is allowed without loss of the current values). In addition to ~~successive~~ sequential versions there are alternative versions. It frequently occurs that a user, after having traced back the chain of execution of a process, desires to make a few changes to the previous state of the data. In these instances the versioning mechanisms allow the taking into account of alternatives or of branches of development that authorize several possible continuations from the same state of the base. An advanced system of traceability should therefore integrate this aspect, all the more since a new branch allows the preceding ones not to be destroyed, thus preserving the traceability of previous processes. There are numerous works that take into account the data whose values develop in time. The domain of time-division databases clearly distinguishes the axis of the validity time from that of the transaction time. The validity time allows, ~~e.g.~~ *e.g.*, the fact to be specified that a price is valid from one date to the next. This information is totally independent of the date of the updating of the data that stores it in the base and that is situated in the time called transactional. By virtue of the specific nature of their problems, the mechanisms for taking account of the validity time comprise solutions of querying and of updating

(publication of R. Snodgrass, "The Temporal Query Language Tquel", ACM Transactions on Database Systems, Association for Computer Machinery, New York, USA), propose operators dedicated to taking account of intervals (between, before, etc.), and specifically treat the cases of updating time intervals for a data that imply a merging or a division (EP 0 984 369 (Fujitsu)). Moreover, the representation and the displaying of different versions require for their part specific solutions (WO 92/13310 (Tandem Telecommunications Systems)) that facilitate the understanding of the development of individual data without being concerned with branches or of the global criterion of the collective ~~coherence~~ consistency of the data of the base in the versioning space. In fact, these aspects are located outside of the problem of traceability, that has a number of requirements relating to versioning that are specific to it, and are still unresolved. Archiving and restoration are finally cited as mechanisms allowing the retrieval of previous states of the database. It is evident on the other hand that they are inadequate faced with the problem of traceability for reasons of too great a granularity in development ~~follow-up~~ tracing, which creates insoluble disadvantages of response time and of storage space. In conclusion, versioning is also indispensable for ensuring traceability but still remains, as will be seen further below, insufficient.

A third level of traceability is that of operations. Tracing an operation amounts to allowing a persistent trace of the execution of this operation, permitting an even better understanding of the manner of how the data develops. In this manner the development of a command between two versions can be better explained if it is known, ~~e.g.~~ *e.g.*, that there was a recovery operation for the total price. The majority of DBMS have ~~journaling~~ logging mechanisms that authorize the consultation of operations carried out at the elementary level. This information should be correlated with the high-level operations so that it can be understood by the users. The basic problem here is that the ~~journal~~ log entries do not have the same persistence cycle as the data. Thus, the ~~journal~~ log is generally located outside of the database and is regularly purged by the administrator. WO 02/27561 (Oracle) brings an alternative solution to this problem by proposing the internal storage (in the database) of transactions and of information about the cancellation of their effects (undo), which allows every previous state of the database to be retrieved by executing in the inverse order the inverse of the operations that took place afterwards. Although interesting, this technique can be very cumbersome in terms of execution time because, to retrieve a precise version of a data, it undoes all the operations that took place afterwards, including those that do not concern it. Moreover, it is not

appropriate either for obtaining the list of all the versions of a data. Finally, it prevents any updating from a previous state of the base, which separates the variants and the alternative branches of development. As will be seen later, ~~this invention adopts~~ I adopt the opposite strategy: Upon receipt of a request, ~~in this invention,~~ its transformation is proceeded to then to an execution of the versioned data. Finally, note the necessity of having information of a higher level supplied, ~~e.g.~~ *e.g.*, by the applications to obtain a connection between the semantics of the applications (application of a recovery upon a command) and that of the DBMS (updating of the attribute "amount" of the command).

The most advanced level of traceability is that of the causality. It concerns the materialization of the links for transporting information at the most elementary level (the finest grain). For example, if any operation O proceeds to read attribute A of data X, to read attribute B of data Y, to the addition of the two and to the storage of the value obtained in this manner in attribute C of data Z, a causal link would be capable of reconstituting this transport of information through the different versions of the data X, Y and Z as well as to the various executions of operation O. This valuable information allows an understanding of the details of the developments and to transitively explain the origins of the modifications and detect the operations to be redone in case of a development of the original data. It is especially important because, contrary to the techniques of ~~journaling~~ logging, it rids itself of the sequential constraint of operations to concentrate on the dynamic dependencies caused by the causality. It is thus possible to become free of, ~~e.g.~~ *e.g.*, thousands of operations, that do not interfere with the data that interests us. Finally, it turns out to also be extremely valuable for simplifying the merging of data located in different branches and for better identifying the true conflicts.

[0037] A particular case of development operation concerns the development of the ~~scheme~~ schema consisting of making the data structure develop without loss of information (Roddick 93 – publication "A Taxonomy for Schema Versioning Based on the Relational and Entity Relationship Models", J.F. Roddick, N.G. Craske and T.J. Richards, 1993). In a manner analogous to that of data, the ~~follow-up~~ tracing of the development of its structure is better ensured if the mechanism of versioning the ~~follow-up~~ tracing of operations and causal traces also applies to the information describing the structure. Particular measures of organizing data and metadata (publication

"Extracting Delta for Incremental Data Warehouse Maintenance", P. Ram et al., Data Engineering, 2000) will be necessary.

[0038] ~~This invention provides~~ I provide a low-intrusive and progressive process for organizing a digital database in a traceable form. ~~The inventor provides~~ I provide the ~~successive~~ sequential levels of traceability described above without, nevertheless, imposing a re-development of existing applications.

[0039] In other words, ~~the invention~~ my method supplies computer applications and their users with the ability to precisely ~~follow~~ trace data along its development by tracing their histories in a complete manner both at the individual level (intermediate versions and successor links) and at the collective level (trigger events and dynamic interdependence links from interactions among the data versions) by positioning it in the coherent framework of its original development.

[0040] It is thus a matter of supplying causality links to an elementary level at which it is possible to readily ~~follow~~ trace the causal stream of transformations and verify the validity of each intermediate operation under the input database of the treatment applied and of the resulting data in such a manner that the reconstitution of every state in the past is immediate.

[0041] In addition, ~~the~~ my process ~~in accordance with the invention~~ makes use of a flexible architectural framework with the least possible amount of constraint and intrusion to supply a very broad applicability to the process proposed and the greatest possible compatibility with the processes of storage and manipulation of the current data.

[0042] To ensure ~~the follow up~~ tracing of the development of a database called "main", ~~the~~ my process ~~of the invention~~ allows one to proceed in such a manner that it represents not only one but all the necessary coherent, ~~successive~~ sequential and/or alternative states of the real world represented in its development while preserving the ACID properties.

[0043] To this end the architecture implemented ~~for the invention~~ is illustrated in Fig. 2 and is constituted as follows:

A ~~journal (J)~~ log (L) organized in the form of an "internal historical database" constituted of a table or a set of tables dedicated to ~~following up~~ tracing the development and based on a mode of universal storage with a stable ~~scheme~~ schema (independent of the logical representation of the applicative data) and particularly adapted to reconstituting data on the fly.

A monitor of transactions (M) and events capable of detecting every request for the development of values and structure transmitted to the database that progressively adds into the dedicated ~~journal~~ log the entries characterizing the elementary development of data (identity, attribute, value, trigger event and dynamic dependencies).

A module for the reconstitution (R) on the fly of the state of the database according to a target event. The system is provided to this end with a cursor (C) dedicated to the selection of the sought state.

One example is: It can be useful to materialize the view of the base called "current" or "main" in the form of tables of specialized structure, ~~e.g.~~ *e.g.*, to permit elevated performances and total compatibility with the existing applications (especially to permit the use of stored procedures and other triggers that an application might need to function correctly).

[0044] The architecture optionally also comprises:

A system for ~~the follow-up of~~ tracing the conformity (SC) of applications with the states of the base and of its ~~scheme~~ schema,

Automatic inoculation tools (I) in the applications of instructions dedicated to the ~~follow-up~~ tracing of dynamic dependencies (capture of data ~~streams~~ flows).

[0045] The ~~journal (J)~~ log (L) of events (or the internal historization database) is constituted primarily of a table with a structure independent of that of the applicative data. The columns are:

A unique identifier of the recording of the logical table concerned by the ~~journal~~ log line belonging to the main key,

A universal event identifier incremented automatically and also belonging to the main key of the ~~journal~~ log and corresponding to the state of the main base,

A value field dedicated to the storage of values.

[0046] The role of the monitor (M) is to detect and correctly interpret each development request while adding the corresponding information into the ~~journal~~ log of events ~~(J)~~ (L).

Kindly replace paragraph [0048] with the following:

[0048] Such a request is processed as follows:

- Syntactic analysis (parsing)of the request,

- Recovery from the ~~scheme~~ schema of identifiers for the client table (53) as well as for the attributes "no_client" (1) [that is, "No_client = client number] and "name_client" (2),

Kindly replace paragraph [0050] with the following:

[0050] Such a request is processed as follows:

- Syntactic analysis (parsing)of the request,

- Recovery from the ~~scheme~~ schema of identifiers for the client table (53) as well as for the attribute "no_client" (1),

- Recovery of the identifier of the recording of the ~~journal~~ log with the value 1001 for attribute No. 1,

- Insertion into the ~~journal~~ log of the last line (using the code 0 for the value).

Examples of development of ~~scheme~~ a schema

Create table client (no_client int primary key)

Creation of a new table	ID	Attribute	UEID	Value	Comments
	53	0	252	8	ID table of the tables
	53	1	253	"client"	Table name
Adding of an attribute	54	0	254	9	Name of attribute
	54	1	255	"no_client"	Name of attribute
	54	2	256	Int	Domain
	54	3	257	PK	Primary key
	54	4	258	53	ID table

Alter table client drop column no_client

- Deletion of an attribute	54	0	278	0	Deletion code

Drop table client

- Deletion of a table	54	0	293	0	Deletion code
Other cases: shifting of attribute	54	3	308	22	Update ID table

Kindly replace paragraphs [0052]-[0055] with the following:

[0052] Each event that tends to modify the logical database finishes by creating one or several entries in the form of new lines (or recordings) in the ~~journal~~ log. This guarantees that nothing is lost and that every logic deletion or updating is not translated into a physical deletion. Thus, the data of the past can be recovered. One of the advantages of this organization is the competing constitution of views such as the books of account that generally block update access by other users.

[0053] Note also the uniformity of the structure for the storage of information: The data is in fact stored in an identical manner whether the development of values or that of the structures is concerned. That is to say that from the viewpoint of logic, it is possible to reconstitute the logic tables as well as their structures on the base of one and the same mechanism. Moreover, the fact of including the ~~journal~~ log in the same database as the main base allows the guaranteeing of its relative ~~coherence~~ consistency by the transactional mechanism assured by the DBMS.

[0054] The reconstitution module (R) is in charge of reconstituting data in a logical format as a function of a parameter of the event type from the ~~journal~~ log of events ~~(J)~~ (L).

[0055] For example, consider that the application wishes to obtain the data from the client table as it was precisely at the time of event 854. This implies selecting event 854 in advance by the event cursor (C). Subsequently, the request "select * from client" is transmitted to the DBMS but transformed by the module (R) into a more complex request obtained in the following manner:

Reconstitution of the corresponding ~~scheme~~ schema: The request relates to the client table. The system must therefore verify the existence of the client table at the historical moment positioned

by the target event and recover the attributes of this logic table (an optimization is possible by keeping the ~~scheme~~ schema in cache),

Recovery of the recordings whose field attribute = 0 created and not deleted "before" the event corresponding to the target state (value = 0 for the deletion code) and attached to this table. In the case of alternatives, "before" only concerns the events located on the same branch,

Recovery of all the recordings of which the field attribute <> 0 attached to the ones preceding and previous to the target event,

Reorganization of the stream of the stored data and grouping by logical recording, that is, in our case by client.

Kindly replace paragraphs [0057] and [0059] with the following:

[0057] In addition to values and events, the ~~journal~~ log can collect invocations of operations. This can be realized by the representation of operations in the form of logic tables in which each operation corresponds to a logic table name and each argument corresponds to a logic attribute. By applying this correspondence ~~scheme~~ schema, the application can send to the ~~journal~~ log (~~e.g.~~ *e.g.*, via an API (application programming interface)) the information necessary for the traceability of operation calls in a manner analogous to the manipulation of logic data (but this task can be automated and given to a post-processor, compiler, processor or even to the virtual machine.

Add (2, 8)

Invocation of the operation Add with the arguments 2 and 8	ID	Attribute	UEID	Value	Comments
57 is the identifier of the operation "add"	62	0	401	57	ID operation "Add"
	62	1	402	2	First argument
62 is the identifier of this invocation of the operation "add"	62	2	403	8	Second argument
	62	999	404	10	Return value

[0058] The operation calls allow linking the semantics of actions of the application to the events recorded in the ~~journal~~ log. As will be seen later, this facilitates positioning of the cursor on the marks significant from the user's viewpoint.

[0059] In addition, the validation points of transactions can be traced in the form of operations. In fact, it is recommended that the cursor be positioned exactly on these points and not between two operations of the same transaction. The ~~coherence~~ consistency of the results depends on this. On the other hand, applications such as the tools aiding in design can benefit greatly from the intermediary states, considered incoherent, for explanatory reasons and also benefit from mechanisms of the "long transactions" type.

Kindly replace paragraphs [0061]-[0063] with the following:

[0061] ~~The invention~~ My process also relates to the materialization of causality links.

[0062] The stream of causal dependencies should be constituted dynamically by reading operations and updated respecting the following rules:

The manipulation of data should systematically consider along with the data read their references of origin and transport it along the stream of data and control. The application should therefore take charge of this aspect by adding to each instruction of manipulation its equivalent of the transport of references, e.g., via an API. The automation of this task can be realized by a post-processor and/or by extensions of the processor or of the virtual machine.

During the insertion of physical data the references of the stream that fed it should be stored in the form of a list of elements of the ID-attribute-UEID type alongside the attribute *value* and this should take place for each physical recording of the ~~journal~~ log. The following table illustrates this. An empty list would correspond to the introduction of a value from outside the system (e.g., by the entry made by a user via an interface-human machine).

ID	Attribute	UEID	Value	Sources	Comments
110	2	54 3	"aaa"		
110	3	54 4	2		

110	4	75 3	"aaa2'	ID	Attribute	UEID	The value of attribute 4 was constituted from attributes 2 and 3
				11 0	2	543	
				11 0	3	544	

[0063] The implementation of sources in the ~~journal~~ log can be realized very well by an additional ~~journal~~ log (or sub-table) organized in a tabular manner for reasons of optimization of performances according to the techniques in effect in the discipline of databases.

Kindly replace paragraph [0067] with the following:

[0067] It is useful to minimize the constants, that is to say the values entered "arbitrarily" so that this information about traceability is entirely effective for the user. The application should therefore give special weight to systems of identification by list selection, pointing, dragging-moving, etc. or by any other technique that simultaneously improves the ergonomics of the application and implicitly allows the ensuring of a ~~follow-up~~ tracing without discontinuity of the information stream. In reality, these techniques are wide-spread because they ensure advantages of static referencing provided in the databases in a current manner.

Kindly replace paragraph [0069] with the following:

[0069] The automatic notification of "recoveries in cause" can be put in place on the base of information about the validity of the data versions in relation to the streams. Thus, for an operation a class of operation, a target or a given source, beacons of ~~coherence~~ consistency of stream can notify the applications by synchronous or asynchronous messages.

Kindly replace paragraph [0071] with the following:

[0071] ~~The~~ My process ~~of the invention~~ is especially designed for managing in an operational manner the historical with the current and the restoration on the fly. Moreover, the managing of storage volumes is facilitated and optimized by a number of factors:

Only the attribute values that change are stored (redundancy is therefore minimized).

The volumes necessary for supplementary storage increase in a linear manner with the number of attributes modified or deleted and do not depend on the data volumes inserted into the base. This factor allows a very advantageous use for a very broad spectrum of applications.

Finally, very pertinent purges can be made according to the data marked as recovered in cause by the traceability links of the source-destination type but this operation should be piloted by the applications as a function of the semantics of recoveries in cause.

Kindly replace paragraphs [0076] and [0082] with the following:

[0076] The architecture implemented for realizing the ~~invention~~ method can also comprise the following modules:

A system for ~~the follow-up of~~ tracing the conformity (SC) of applications with the states of the base and of its ~~scheme~~ schema. The principle is based on the recording of a version identifier of the application to declare a level of compatibility with the state or states corresponding to the ~~scheme~~ schema of the main base,

Tools for automatic inoculation (I) into the applications of instructions dedicated to the ~~follow-up~~ tracing of dynamic dependencies (capture of data ~~streams~~ flows): pre-post-processor or expanded virtual machine,

Visual components specialized in the navigation and exploration of the base states (not shown).

[0077] The ~~invention~~ method can be implemented in several ways in accordance with the context in which it is integrated in an application.

[0078] Fig. 3 shows an architecture that permits three levels of integration of traceability from bottom to top:

The existing applications can continue to access the database (called "main") in the same manner. The base can either retain its original structure and redirect the access to an associated ~~journal~~ log (called internal base), or develop toward a physical organization of the ~~journal~~ log

type and offer views or a driver in charge of the translation of requests and results.

[0079] Existing applications can be readily provided with a "cursor" on the condition that the access to the data is centralized (which is generally the case, e.g., via a single driver). In this instance, the application can offer automatic access means to the databases (now implemented in the form of a ~~journal~~ log) and permit users to actuate a cursor that positions the readings on the desired event mark. Slight adaptations can take place to reconcile the granularity of the events with the semantics of the application.

[0080] New applications constructed entirely on the base of the technologies of the inoculation of the generation of traces benefit implicitly from the most advanced level of traceability offered by this process comprising an exhaustive ~~follow-up~~ tracing of the development of data and of their structure. It is sufficient to return to the declarative techniques of the representation of sources, to commit them to the same ~~journal~~ log and to have them manipulated by an assembly tool provided itself with a traceability module in accordance with this process such that the ~~follow-up~~ tracing of the development of applications is ensured at the same level.

[0081] This architecture permits the gradual attainment of more and more elevated levels of traceability of persistent data:

Initial: Representation and persistence (indispensable, previous), ensured by the initial persistence system

~~Journalization~~ Logging of events (useful, short-term recovery in case of breakdown but poses a problem of rapid reconstitution of past states

Historical and versioning (useful because the values stored are multiple and can comprise variants, but this functionality generates problems of reconstitution in a mode compatible with the initial mode)

Structural development: The ~~follow-up~~ tracing of development of data and of the ~~scheme~~ schema of the main database, compatible with the initial mode

Causal ~~dependence~~ dependency: The detection of ~~streams of~~ dynamic ~~dependence~~ dependency flow and causality links between the data of the historical database (~~journalized~~ logged).

[0082] The use of branches offers the possibility of creating alternatives of development of the database. At the same time, this raises new problems regarding traceability. In fact, suppose that

after the separation of branches A and B, data X is modified in branch A by operation O. It may then be desirable to send its new value to branch B as if it had had this value at the moment of the separation of the branches. This operation, called "refreshing," is very useful for numerous instances in which institutional reference data is received at more or less regular intervals. Their integration can then pose problems of interference with the operations carried out in the meantime. For example, if no operation that had as source or destination data X in branch B was performed in the meantime, it can be considered that there is no impact. On the other hand, if that is the case, it is then necessary to decide (explicitly or implicitly) which operation has priority and to redo the others. These conflicts are readily detectable by the links of dynamic ~~dependence~~ dependency. The associated semantics will be supplied by that of the operations that caused these dependencies. A simple comparison of the universal identifier of the traces of operations allows the evaluation of priority and to confirm it or cancel it. The user (or the application via a system of predefined rules) can thus decide with knowledge. The case of a merging of branches is quite analogous.

Kindly replace paragraphs [0087] and [0088] with the following:

[0087] The spectrum of applications ~~of the invention~~ covers the majority of cases in which it is useful to ~~follow~~ trace the development of persistent data, management applications and up to file management systems using design tools based on universal sets (or repository), or beyond the requirements of persistence if the ~~follow-up~~ tracing of the development is useful.

[0088] Although this ~~invention~~ method has been described in connection with specific forms thereof, it will be appreciated that a wide variety of equivalents may be substituted for the specified elements described herein without departing from the spirit and scope of this ~~invention~~ method as described in the appended claims.